Introduction to Provable Security in Public-Key Cryptography

Damien Vergnaud
(Mathematical Foundations of Asymmetric Cryptography)

Sorbonne Université – CNRS – IUF

Contents

Cryptography

Goal: enable "secure" communication in the presence of adversaries

internet, phone line, ...

eavesdrops

Alice

Bob

Eve

Encryption

Alice sends a ciphertext to Bob
Only Bob can recover the plaintext

CONFIDENTIALITY

To recover the plaintext

- to find the whole plaintext ?
- to get some information about it ?

Which means can be used ?

- just the ciphertext ?
- some extra information ?

Why “Provable Security” ?

Once a cryptosystem is described, how can we prove its security?

by trying to exhibit an attack

- attack found
  $\Rrightarrow$ system insecure!
- attack not found
  $\Rrightarrow$?

by proving that no attack exists under some assumptions

- attack found
  $\Rrightarrow$ false assumption

- **"Textbook" cryptosystems cannot be used as such**
- **Praticioners need formatting rules to ensure operability.**
  $\rightsquigarrow$ Paddings are used in practice : heuristic security
- **Provable security is needed in upcoming systems.**
  This is no longer just theory.
- **Provable security is fun!** :-)

Who is the bad guy?

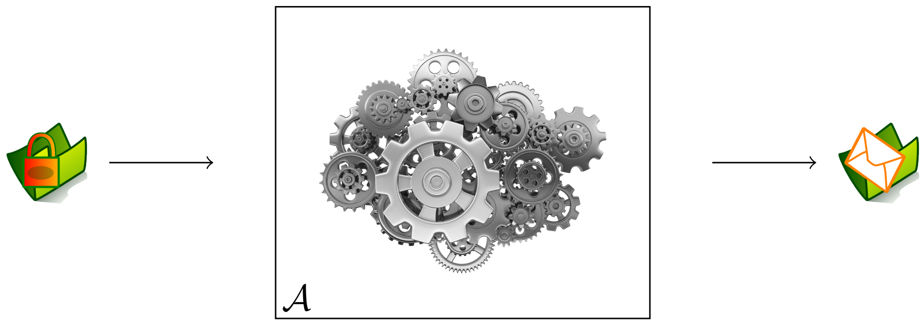
We are protecting ourselves from the evil Eve, who

- is a probabilistic polynomial time Turing machine (PPTM) **(Church-Turing thesis)**
- knows all the algorithms **(Kerckoff's principles)**
- has full access to communication media.

Proof by reduction

$\mathcal{A}$ adversary against e.g. one-wayness

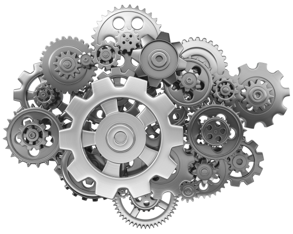
Proof by reduction



$\mathcal{A}$ adversary against e.g. one-wayness

Proof by reduction

Instance $\mathcal{I}$ of a problem $\mathcal{P}$

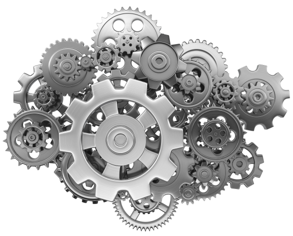


$\mathcal{A}$

Proof by reduction

Instance $\mathcal{I}$ of a problem $\mathcal{P}$

$\downarrow$



$\mathcal{A}$

$\mathcal{R}$

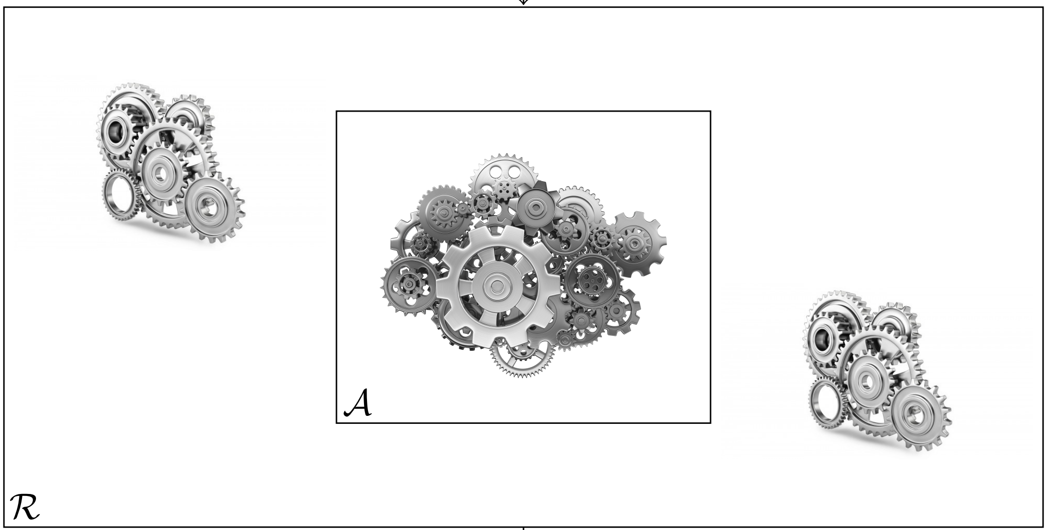
$\downarrow$

Solution of $\mathcal{I}$

Proof by reduction

Instance $\mathcal{I}$ of a problem $\mathcal{P}$

$\downarrow$



$\downarrow$

Solution of $\mathcal{I}$

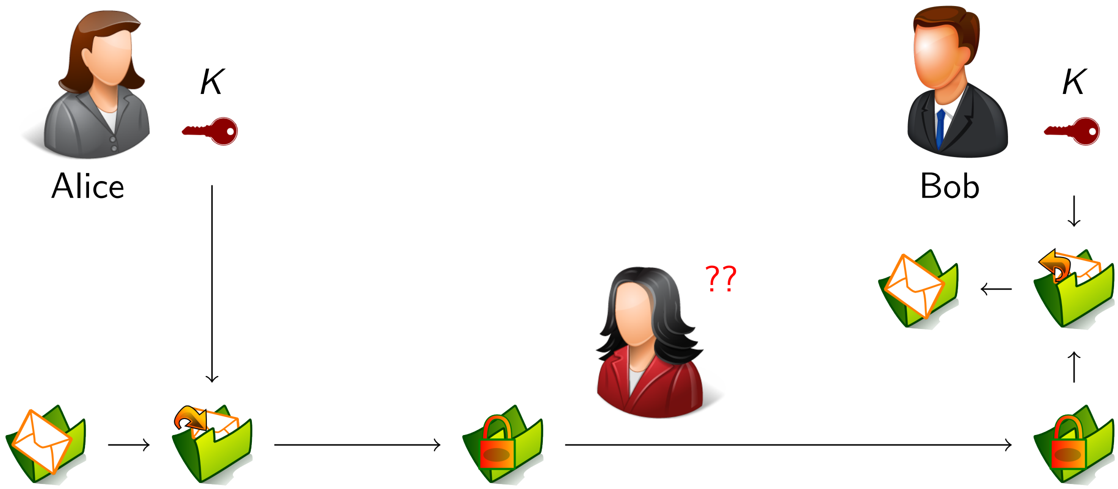
$\mathcal{P}$ intractable $\rightarrow$ contradiction

The Methodology of "Provable Security"

1. Define goal of adversary
2. Define security model
3. Define complexity assumptions
4. Provide a proof by reduction
5. Check proof
6. Interpret proof

Secret-Key Encryption

Symmetric encryption: Alice and Bob share a "key" K



- Bob can use the same method to send messages to Alice.
  $\rightsquigarrow$ **symmetric setting**
- How did Alice and Bob establish K?

The solution: Public-Key Cryptography

- first proposed by Diffie and Hellman:

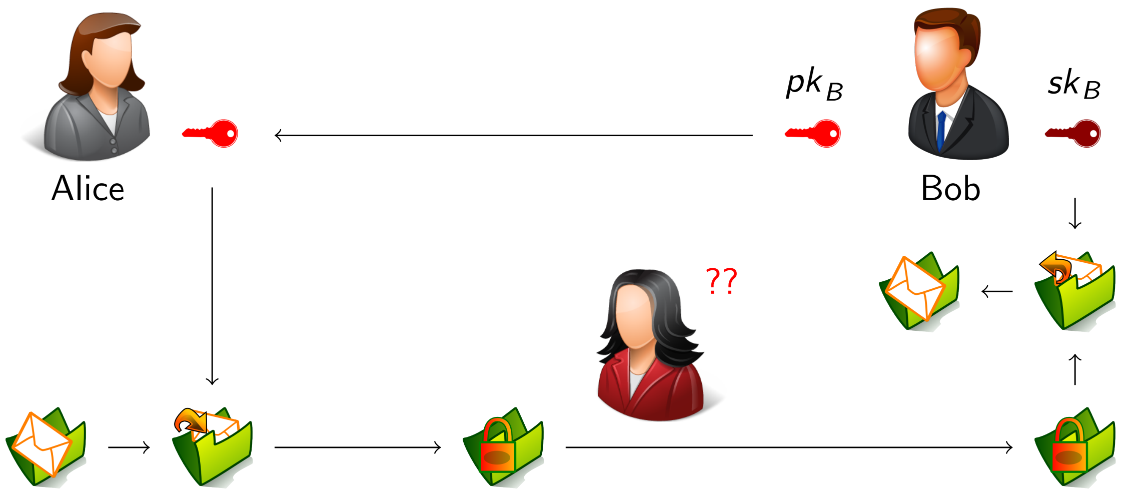
W.Diffie and M.E.Hellman,
New directions in cryptography
IEEE Trans. Inform. Theory, IT-22, 6, 1976, pp. 644-654.

- 2015 Turing Award
- It 1997 the GCHQ revealed that they new it already in 1970 (James Ellis).

Public-Key Encryption

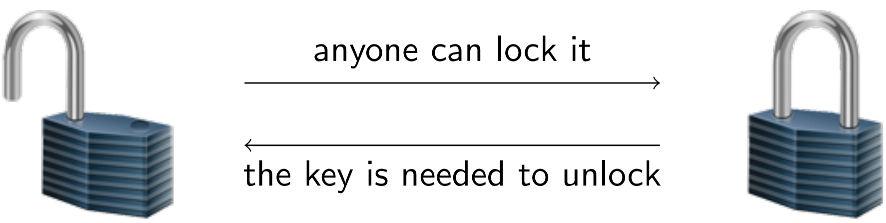
Asymmetric encryption: Bob owns two “keys”

- a public key — known by everybody (including Alice)
- a secret key — known by Bob only



But is it possible?

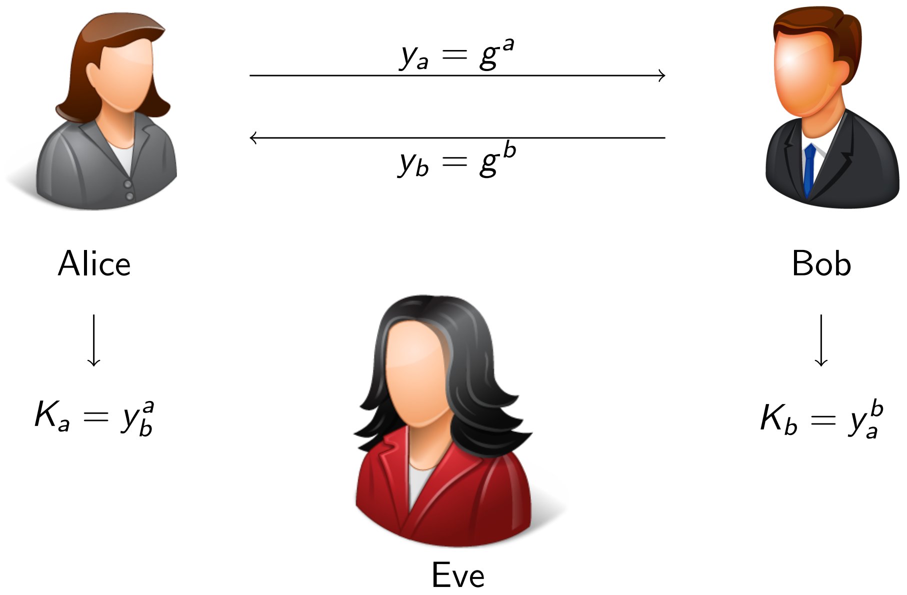
- In “physical world”: yes!
  $\rightsquigarrow$ Example: padlock



- Diffie and Hellman proposed the public-key cryptography in 1976.
  - They just proposed the concept, not the implementation.
  - But they have shown a protocol for key-exchange

Diffie-Hellman Key Exchange

$(\mathbb{G}, \cdot)$ a finite cyclic group; $\langle g \rangle = \mathbb{G}$



$$K_a = y_b^a = (g^b)^a = g^{ab} = (g^a)^b = y_a^b = K_b$$

Diffie-Hellman Key Exchange: Security

Eve knows:

- $(\mathbb{G}, g)$
- $y_a = g^a$
- $y_b = g^b$

and should have "no information" on $K = g^{ab}$.

- If finding a from y_a is easy then the DH key exchange is not secure.
- Even if it is hard, then

  ...the scheme may also not be completely secure

- How to choose the group $\mathbb{G}$?

  see Pierrick's lectures

- Do we really need a group?

  see Luca's lectures

Public-Key Encryption

An asymmetric encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where

- $\mathcal{K}$ is a probabilistic **key generation algorithm** which returns random pairs of secret and public keys (sk, pk) depending on the security parameter κ,
- $\mathcal{E}$ is a probabilistic **encryption algorithm** which takes on input a public key pk and a *plaintext* $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns a *ciphertext* c,
- $\mathcal{D}$ is a deterministic **decryption algorithm** which takes on input a secret key sk, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$.

If $(sk, pk) \leftarrow \mathcal{K}$, then $\mathcal{D}_{sk}\left(\mathcal{E}_{pk}(m, u)\right) = m$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Encryption: Security Notions

Encryption is supposed to provide confidentiality of the data.

But what exactly does this mean?

Security goal	But ...
Recovery of secret key is infeasible	True if data is sent in the clear
Obtaining plaintext from ciphertext is infeasible	Might be able to obtain half the plaintext
etc	*etc*

So what is a **secure** encryption scheme ?

Not an easy question to answer . . .

Attackers should not be able to compute any information about m.

How to formalize it ?

Attackers should not be able to compute any information about m.

Probabilistic approach

- M some random variable that takes values from $\mathcal{M}$
- K random variable distributed uniformly over $\mathcal{K}$
- $C = \mathcal{E}_K(M)$

Definition

An encryption scheme is perfectly secret if for every random variable M and every $m \in \mathcal{M}$ and every $c \in \mathcal{C}$ with $\Pr(C = c) > 0$:

$$\Pr(M = m) = \Pr(M = m | C = c)$$

$\rightsquigarrow$ C and M are independent

A perfectly symmetric secure scheme: **one-time pad**

Description

- $\ell \in \mathbb{N}$ a parameter. $\mathcal{M} = \mathcal{K} = \{0,1\}^\ell$.
- Let $\oplus$ denote component-wise XOR.
- **Vernam's cipher:** $\mathrm{Enc}(K, m) = m \oplus K$ and $\mathrm{Dec}(K, c) = c \oplus K$.

- One-time pad is perfectly secret!

$$\begin{aligned} \Pr(C = c | M = m) &= \Pr(K \oplus M = c | M = m) \\ &= \Pr(K = m \oplus c | M = m) = 2^{-\ell} \end{aligned}$$

- Each key cannot be used more than once!

$$\mathrm{Enc}(K, m_0) \oplus \mathrm{Enc}(K, m_1) = (m_0 \oplus K) \oplus (m_1 \oplus K) = m_0 \oplus m_1$$

- One time-pad is optimal in the class of perfectly secret schemes

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally defines a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **model**).

A security notion (or level) is entirely defined by pairing an adversarial goal with an adversarial model.

Examples: OW-PCA, IND-CCA2, NM-CCA2.

History of Security Goals

- it shouldn't be feasible to compute the secret key *sk* from the public key *pk* (unbreakability or UBK). Implicitely appeared with public-key crypto.
- it shouldn't be feasible to invert the encryption function over any ciphertext under any given key *pk* (one-wayness or OW). Diffie and Hellman, late 70's.
- it shouldn't be feasible to recover a *single bit of information* about a plaintext given its encryption under any given key *pk* (semantic security or SEM). Goldwasser and Micali, 1982.
- it shouldn't be feasible to distinguish pairs of ciphertexts based on the message they encrypt (indistinguishability or IND). Goldwasser and Micali, 1982.
- it shouldn't be feasible *to transform* some ciphertext into another ciphertext such that plaintext are meaningfully related (non-malleability or NM). Dolev, Dwork and Naor, 1991.

History of Adversarial Models

Several types of computational resources an adversary has access to have been considered:

- **chosen-plaintext attacks** (CPA), unavoidable scenario.
- **non-adaptive chosen-ciphertext attacks** (CCA1), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext.
  Naor and Yung, 1990.
- **adaptive chosen-ciphertext attacks** (CCA2) as a scenario in which the adversary queries the decryption oracle before and *after* being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself.
  Rackoff and Simon, 1991.

Semantic Security

Semantic security for $\mathcal{E} = (G, E, D)$, against an adversary $\mathcal{A}$ and attack $atk \in \{\text{cpa}, \text{cca1}, \text{cca2}\}$ is measured using the following game:

Experiment $\textbf{Expt}_{\mathcal{E}}^{\text{sem-}atk\text{-}b}(\mathcal{A}, \kappa)$:
$(pk, skK) \leftarrow G(1^\kappa)$;
$(\mathcal{M}, s) \leftarrow \mathcal{A}^{D_0(\cdot)}(\text{select}, pk)$;
$x_0 \overset{R}{\leftarrow} \mathcal{M}$; $x_1 \overset{R}{\leftarrow} \mathcal{M}$;
$y \leftarrow E_{pk}(x_b)$;
$(f, \alpha) \leftarrow \mathcal{A}^{D_1(\cdot)}(\text{predict}, y, s)$;
if $f(x_b) = \alpha$ **then return** 1;
else return 0;

- $\mathcal{M} : \mathcal{P} \to [0, 1]$ is a **distribution** over the plaintext space
- $f : \mathcal{P} \to \text{ran} f$ is a **function** on plaintexts, with $\alpha \in \text{ran} f$.
- The **oracles** D_0 and D_1 are defined according to *atk*:

atk	$D_0(x)$	$D_1(x)$
CPA	$\perp$	$\perp$
CCA1	$D_{sk}(x)$	$\perp$
CCA2	$D_{sk}(x)$	$D_{sk}(x)$ for $x \neq y$

Indistinguishability

Indistinguishability for $\mathcal{E} = (G, E, D)$, against an adversary $\mathcal{A}$ and attack $atk \in \{\text{cpa}, \text{cca1}, \text{cca2}\}$ is measured using the following game:

Experiment $\mathbf{Expt}_{\mathcal{E}}^{\text{ind-}atk\text{-}b}(\mathcal{A}, \kappa)$:
$(pk, skK) \leftarrow G(1^{\kappa})$;
$(x_0, x_1, s) \leftarrow \mathcal{A}^{D_0(\cdot)}(\text{find}, pk)$;
if $|x_0| \neq |x_1|$ **then return** 0;
$y \leftarrow E_{pk}(x_b)$;
$b' \leftarrow \mathcal{A}^{D_1(\cdot)}(\text{guess}, y, s)$;
return b';

- In the first stage, the adversary has to choose two plaintexts.
- One is encrypted by the challenger and the ciphertext given to the adversary.
- The adversary must decide which plaintext was encrypted.
- The **oracles** D_0 and D_1 are defined according to atk:

atk	$D_0(x)$	$D_1(x)$
CPA	$\perp$	$\perp$
$CCA1$	$D_{sk}(x)$	$\perp$
$CCA2$	$D_{sk}(x)$	$D_{sk}(x)$ for $x \neq y$

Advantage and insecurity

For a public-key encryption scheme $\mathcal{E}$, under attack $atk \in \{\text{cpa}, \text{cca1}, \text{cca2}\}$ by an adversary $\mathcal{A}$, we define $\mathcal{A}$'s advantage by:

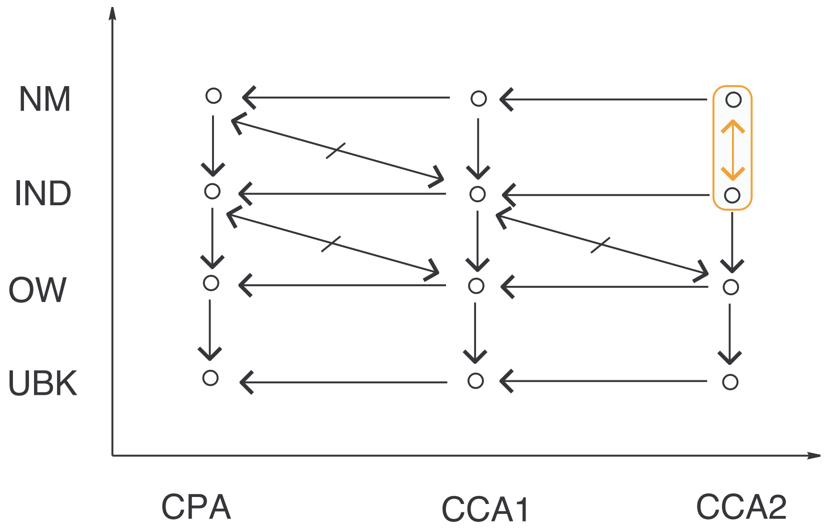
$$\begin{aligned} \text{Adv}^{ind-atk}\mathcal{E}(\mathcal{A}) &= \Pr[\mathbf{Expt}_{\mathcal{E}}^{\text{ind-}atk\text{-1}}(\mathcal{A}) = 1] - \Pr[\mathbf{Expt}_{\mathcal{E}}^{\text{ind-}atk\text{-0}}(\mathcal{A}) = 1]; \\ \text{Adv}^{sem-atk}\mathcal{E}(\mathcal{A}) &= \Pr[\mathbf{Expt}_{\mathcal{E}}^{\text{sem-}atk\text{-1}}(\mathcal{A}) = 1] - \Pr[\mathbf{Expt}_{\mathcal{E}}^{\text{sem-}atk\text{-0}}(\mathcal{A}) = 1]. \end{aligned}$$

We define insecurities for $goal \in \{\text{ind}, \text{sem}\}$ under chosen plaintext attacks, and chosen ciphertext attacks $cca \in \{\text{cca1}, \text{cca2}\}$ by:

$$\begin{aligned} \mathbf{InSec}^{goal\text{-cpa}}(\mathcal{E}; t) &= \max_{\mathcal{A}} \text{Adv}^{goal-cpa}\mathcal{E}(\mathcal{A}); \\ \mathbf{InSec}^{goal\text{-}cca}(\mathcal{E}; t, q_D) &= \max_{\mathcal{A}} \text{Adv}^{goal-cca}\mathcal{E}(\mathcal{A}). \end{aligned}$$

where the maxima are taken over adversaries $\mathcal{A}$ which run in time t and issue q_D decryption queries.

Relations Among Security Notions



Chosen-Ciphertext Security

Because IND-CCA2 $\equiv$ NM-CCA2 is the upper security level, it is desirable to prove security with respect to this notion. It is also denoted by IND-CCA and called chosen ciphertext security.

- Formally, an asymmetric encryption scheme is said to be (τ, ε)-IND-CCA if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time upper-bounded by τ,

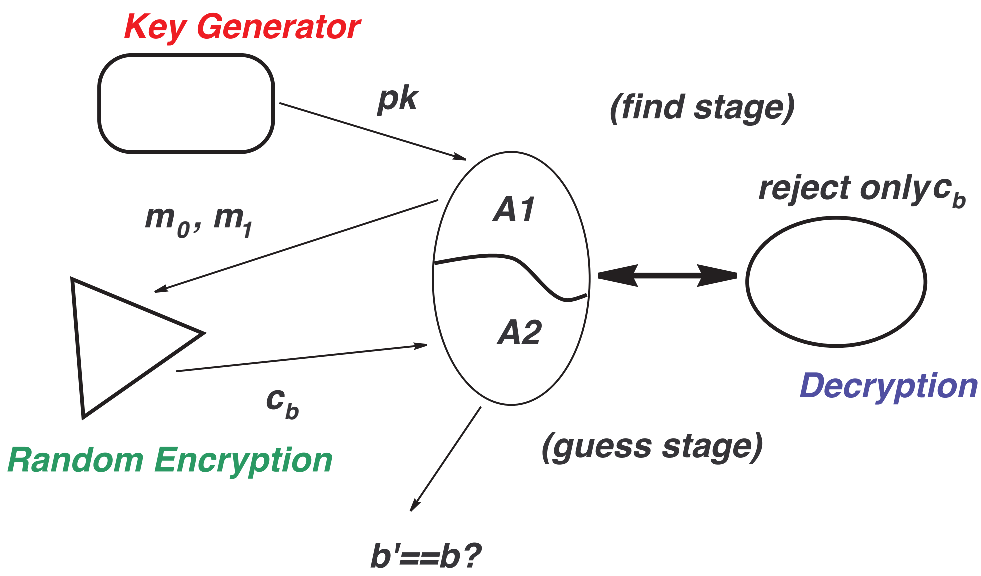
$$\mathsf{Adv}^{\mathsf{ind}}\mathcal{E}(\mathcal{A}) = 2 \times \Pr_{\substack{b \xleftarrow{R} \{0,1\} \\ u \xleftarrow{R} \mathcal{U}}} \left[\begin{array}{l} (sk, pk) \leftarrow \mathcal{K}(1^\kappa), (m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk) \\ c \leftarrow \mathcal{E}_{pk}(m_b, u) : \mathcal{A}_2(c, \sigma) = b \end{array} \right] - 1 < \varepsilon \ ,$$

  where the probability is taken over the random choices of $\mathcal{A}$.

- The two plaintexts m_0 and m_1 chosen by the adversary have to be of **identical length**.

- Access to a **decryption oracle** is allowed throughout the game.

IND-CCA: Playing the Game



The ElGamal public-key encryption scheme

ElGamal's encryption scheme is based on **Diffie-Hellman**. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q.

Plaintexts and ciphertexts in the scheme are elements of G.

The scheme $\mathcal{E}$-*ElGamal* = (G-*ElGamal*, E-*ElGamal*, D-*ElGamal*) is defined by:

G-*ElGamal*:
$\alpha \stackrel{R}{\leftarrow} \{0, 1, \ldots, q-1\}$;
return $(pk = g^\alpha, sk = \alpha)$;

E-*ElGamal*$_{pk}(m)$:
$\beta \stackrel{R}{\leftarrow} \{0, 1, \ldots, q-1\}$;
return $(g^\beta, m \cdot pk^\beta)$;

D-*ElGamal*$_{sk}(y)$:
$(B, C) \leftarrow y$;
$m \leftarrow B^{-sk} C$;
return m;

This scheme is secure in the IND-CPA sense if the Decisional Diffie-Hellman problem is hard in G.

Security proof for ElGamal

Suppose $\mathcal{A}$ is an adversary attacking the ElGamal scheme in the IND-CPA sense.

We construct from it an algorithm $\mathcal{D}$ which solves the DDH problem (i.e., given a triple $A = g^\alpha, B = g^\beta, C$, decides whether $C = g^{\alpha\beta}$):

Algorithm $\mathcal{D}(A, B, C)$:
$(m_0, m_1, s) \leftarrow \mathcal{A}(\text{find}, A)$;
$b \stackrel{R}{\leftarrow} \{0, 1\}$;
$y \leftarrow (B, m_b \cdot C)$;
$b' \leftarrow \mathcal{A}(\text{guess}, y, s)$;
if $b = b'$ **then return** 1;
else return 0;

Security proof for ElGamal (cont.)

Let α and β be the discrete logs of A and B.

- If $C = g^{\alpha\beta}$, then $\mathcal{D}$'s success probability is equal to $\mathcal{A}$'s probability of guessing the hidden bit correctly, which is

$$\frac{\mathsf{Adv}^{ind-cpa}\mathcal{E}\text{-}ElGamal^G(\mathcal{A})}{2} + \frac{1}{2}.$$

- If C is random, then $m_b C$ is uniformly distributed in G, and independent of b, so $\mathcal{A}$ answers correctly with probability exactly $\frac{1}{2}$.

Hence, $\mathsf{Adv}^{ddh}\mathbb{G}(\mathcal{D}) = \mathsf{Adv}^{ind-cpa}\mathcal{E}\text{-}ElGamal(\mathcal{A})/2$, and

$$\textbf{InSec}^{\text{ind-cpa}}(\mathcal{E}\text{-}ElGamal^G; t) \leq 2 \cdot \textbf{InSec}^{\text{ddh}}(G; t).$$

$\square$

Notes about ElGamal

- **We needed the Decisional Diffie-Hellman assumption to prove the security.** This is a strong assumption. Still, a proof based on DDH is a lot better than nothing.

- **We really do need the Decisional Diffie-Hellman assumption.**

  An adversary with a DDH algorithm can submit $m_0 \in_R G$ and $m_1 = 1$; it receives a ciphertext (B, C), and returns 1 if (A, B, C) looks like a Diffie-Hellman triple, or 0 if it looks random.

- **The plaintexts must be elements of the cyclic group $\mathbb{G}$.**

  For example, if $\mathbb{G}$ is a subgroup of $\mathbb{F}_p^*$, it's *not* safe to allow elements outside the subgroup as plaintexts: an adversary can compare orders of ciphertext elements to break the semantic security of the scheme.

- **ElGamal is malleable.** We can decrypt a challenge ciphertext $y = (g^\beta, A^\beta x)$ by choosing a random γ and requesting a decryption of $y' = (g^{\beta\gamma}, A^{\beta\gamma} x^\gamma)$.

Random Oracle Model

- idealized model introduced by Bellare and Rogaway in 1993

- considers cryptographic constructions that make use of a function H
  - can be accessed in a black-box way
  - answers consistently for values x already queried
  - for new values x, choose uniformly at random in the range as answer

- Do they exist?
  $\rightsquigarrow$ NO! But let us assume cryptographic hash functions behave "approximately" like ROs

Random Oracle Model

- **Why ROM?**
  - allows efficient constructions of cryptographic primitives with somewhat "provable security" guarantees
  - Efficient signature and encryption schemes (Schnorr signatures, ...)
- **How are ROs used in security proofs?**
  - Sample a random H at the beginning of an experiment
  - Output of ROM fully hidden unless queried, i.e., $H(m, r)$ for r a large random string
  - Typically we assume that the reduction can "program" the random oracle i.e., can choose the answers to the oracle calls
- **Criticism of the ROM**
  - only a "heuristic" argument for security instead of a real proof
  - There are schemes that can be shown secure in the ROM, but insecure when ROM is replaced with any real hash function

The Hash ElGamal public-key encryption scheme

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of order q.
Let $\mathcal{H} : \mathbb{G} \to \{0,1\}^\ell$ be an hash function.

Plaintexts are elements of $\{0,1\}^\ell$.

G-H-ElGamal:
$\alpha \xleftarrow{R} \{0, 1, \dots, q-1\}$;
return $(pk = g^\alpha, sk = \alpha)$;

E-H-ElGamal$_{pk}(m)$:
$\beta \xleftarrow{R} \{0, 1, \dots, q-1\}$;
return $(g^\beta, m \oplus \mathcal{H}(pk^\beta))$;

D-H-ElGamal$_{sk}(y)$:
$(B, C) \leftarrow y$;
$m \leftarrow \mathcal{H}(B^{sk}) \oplus C$;
return x;

This scheme is secure (in the Random Oracle Model) in the IND-CPA sense if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$.

The Hash ElGamal public-key encryption scheme

Let $G = \langle g \rangle$ be a cyclic group of order q.
Let $\mathcal{H} : G \to \{0,1\}^\ell$ be an hash function.
Let $\mathcal{G} : G \times \{0,1\}^\ell \to \{0,1\}^k$ be an hash function.

Plaintexts are elements of $\{0,1\}^\ell$.

G-H+-ElGamal:
$\alpha \stackrel{R}{\leftarrow} \{0,1,\ldots,q-1\}$;
return $(a = g^\alpha, \alpha)$;

E-H+-ElGamal$_{pk}(x)$:
$\beta \stackrel{R}{\leftarrow} \{0,1,\ldots,q-1\}$;
return $(g^\beta, x \oplus \mathcal{H}(pk^\beta), \mathcal{G}(x, pk^\beta))$;

D-H+-ElGamal$_{sk}(y)$:
$(B, c, d) \leftarrow y$;
$x \leftarrow \mathcal{H}(B^{sk}) \oplus c$;
return x if $d = \mathcal{G}(x, B^{sk})$;
return $\perp$ otherwise.

This scheme is IND-CCA2 (in the Random Oracle Model) if the (strong) Computational DH problem is hard in $\mathbb{G}$.

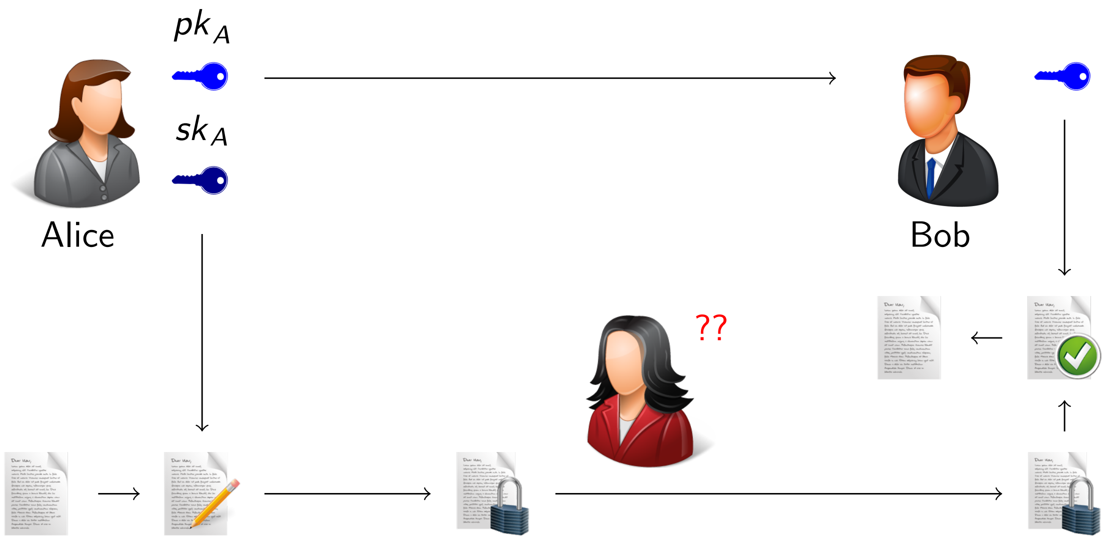
Digital Signatures

- A very important public key primitive is the **digital signature**.
- The idea is
  - Message + Alice's Private Key = Signature
  - Message + Signature + Alice's Public Key = YES/NO
- Alice can sign a message using her private key.
- Anyone can verify Alice's signature, since everyone can obtain her public key.
- the verifier is convinced that only Alice could have produced the signature
  - **only Alice knows her private key!**

Digital signature schemes

Digital signatures: Alice owns two "keys"

- a public key — known by everybody (including Bob)
- a secret key — known by Alice only



Digital Signatures : Services

- The verification algorithm is used to determine whether or not the signature is properly constructed.
- the verifier has guarantee of
  - message integrity and
  - message origin.
- also provide non-repudiation - not provided by MACs.

Most important cryptographic primitive!

Security Notions

Depending on the context in which a given cryptosystem is used, one may formally define a security notion for this system,

- by telling what **goal** an adversary would attempt to reach,
- and what means or information are made available to her (the **attack model**).

A security notion (or level) is entirely defined by pairing an adversarial goal with an adversarial model.

Examples: UB-KMA, UUF-KOA, EUF-SOCMA, EUF-CMA.

Signature Schemes

An digital signature scheme is a triple of algorithms $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ where

- $\mathcal{K}$ is a probabilistic **key generation algorithm** which returns random pairs of secret and verification keys (sk, vk) depending on the security parameter κ,
- $\mathcal{S}$ is a (probabilistic) **signature algorithm** which takes on input a signing key sk and a *message* $m \in \mathcal{M}$, runs on a random tape $u \in \mathcal{U}$ and returns $s \in S$,
- $\mathcal{V}$ is a deterministic **verification algorithm** which takes on input a verification key vk, a message m and $s \in S$ and outputs a bit in $\{0, 1\}$.
  If $\mathcal{V}_{\sqsubseteq\|}(\Updownarrow, \int) = \infty$, then s is a *signature* on m for vk.

If $(sk, vk) \leftarrow \mathcal{K}$, then $\mathcal{V}_{vk}(m, \mathcal{S}_{sk}(m, u)) = 1$ for all $(m, u) \in \mathcal{M} \times \mathcal{U}$.

Security Goals

[Unbreakability] the attacker recovers the secret key *sk* from the public key *vk* (or an equivalent key if any). This goal is denoted **UB**. Implicitly appeared with public-key cryptography.

[Universal Unforgeability] the attacker, without necessarily having recovered *sk*, can produce a valid signature of any message in the message space. Noted **UUF**.

[Existential Unforgeability] the attacker creates a message and a valid signature of it (likely not of his choosing). Denoted **EUF**.

Adversarial Models

- **Key-Only Attacks** (KOA), unavoidable scenario.
- **Known Message Attacks** (KMA) where an adversary has access to signatures for a set of known messages.
- **Chosen-Message Attacks** (CMA) the adversary is allowed to use the signer as an oracle (full access), and may request the signature of any message of his choice

Chosen-Message Security

Goldwasser, Micali, Rivest (1988)
A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks.
SIAM J. Comput. 17(2) pp. 281-308.

Formally, an signature scheme is said to be (q, τ, ε)-secure if for any adversary $\mathcal{A}$ with running time upper-bounded by τ,

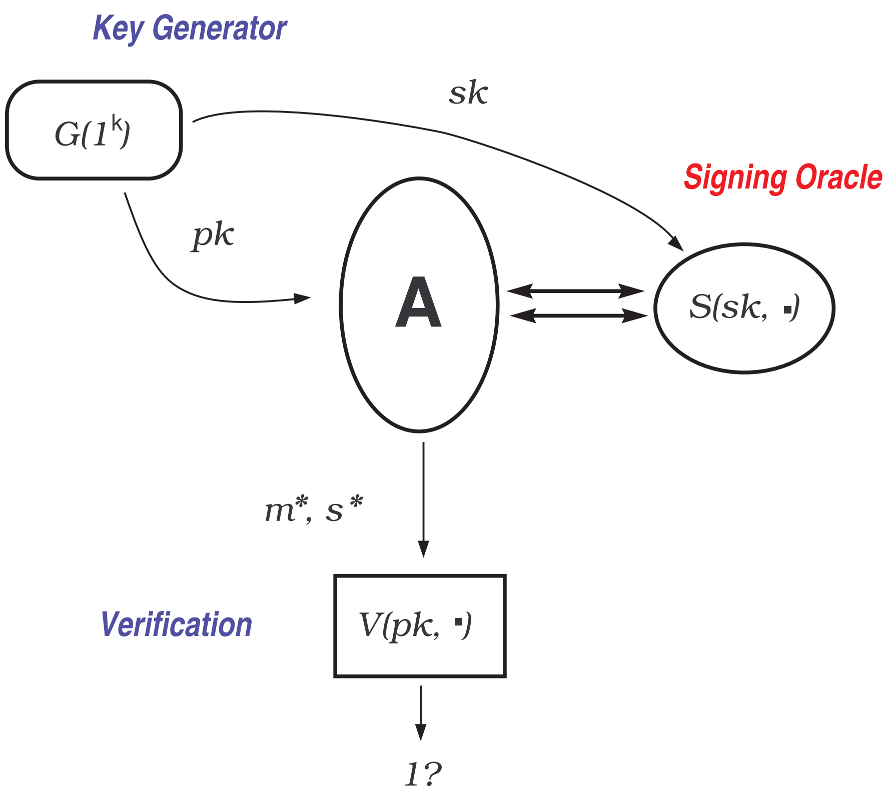
$$\mathsf{Succ}^{\mathsf{EUF-CMA}}(\mathcal{A}) = \Pr\left[\begin{array}{l} (sk, vk) \leftarrow \mathcal{K}(1^k), \\ (m^*, s^*) \leftarrow \mathcal{A}^{\mathcal{S}(sk,\cdot)}(pk), \\ \mathcal{V}(vk, m^*, s^*) = 1 \end{array}\right] < \varepsilon\ ,$$

where the probability is taken over all random choices.

The notation $\mathcal{A}^{\mathcal{S}(sk,\cdot)}$ means that the adversary has access to a signing oracle throughout the game, but at most q times.

The message m^* output by $\mathcal{A}$ was **never** requested to the signing oracle...

EUF-CMA: Playing the Game



Lamport signatures

L. Lamport
Constructing digital signatures from a one-way function
Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, Oct. 1979.

- a Lamport signature or **Lamport one-time signature scheme** is a method for constructing efficient digital signatures.
- Lamport signatures can be built from any cryptographically secure **one-way** function; usually a **cryptographic hash function** is used.
- Unfortunately each Lamport key can only be used to sign a single message.
- However, we will see how a single key could be used for **many** messages.

How to sign **one** bit **just once** ?

$$\boxed{\mathcal{M} = \{0,1\}}$$

- **Key generation:**
  - Consider $f : X \longrightarrow Y$ a **one-way function**.
    e.g.
    $$\begin{array}{rccc} f: & \mathbb{Z}_q & \longrightarrow & \mathbb{G} \\ & x & \longmapsto & f(x) = g^x \end{array} \quad r$$
  - Select two random elements $x_0, x_1 \in X$.
  - Compute their images $y_i = f(x_i)$ for $i \in \{0,1\}$.

  Verification key $vk = (y_0, y_1)$ which can be published.
  Signing key $sk = (x_0, x_1)$ which needs to be kept secret
- **Signature:** if Alice wants to sign a bit b, she does the following:
  - Use her signing key (x_0, x_1) to send the signature $s = x_b$ to Bob.
- **Verification:** to check the validity of s on b, Bob does the following:
  - Obtain Alice's authentic verification key (y_0, y_1).
  - Check whether $f(s) = y_b$.

How to sign **k** bits **just once** ?

$\mathcal{M} = \{0,1\}^k$

- **Key generation:**
  - Generate $f : X \longrightarrow$ a **one-way function**.
  - Select $2k$ random elements $x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k} \in X$.
  - Compute their images $y_{i,j} = f(x_{i,j})$ for $i \in \{0,1\}$ and $j \in [\![1,k]\!]$.

  Verification key $vk = (y_{0,1}, y_{1,1}, \ldots, y_{0,k}, y_{1,k})$ which can be published.
  Signing key $sk = (x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k})$ which needs to be kept secret
- **Signature:** if Alice wants to sign $m = m_1 \ldots m_k$, she does the following:
  - Use her signing key $(x_{0,1}, x_{1,1}, \ldots, x_{0,k}, x_{1,k})$ to send the signature $s = (x_{m_1,1}, x_{m_1,2}, \ldots, x_{m_k,k})$ to Bob.
- **Verification:** to check the validity of $s = (s_1, \ldots, s_k)$ on m, Bob does the following:
  - Obtain Alice's authentic verification key $(y_{0,1}, y_{1,1}, \ldots, y_{0,k}, y_{1,k})$.
  - Check whether $f(s_i) = y_{m_b,i}$ for all $i \in [\![1,k]\!]$.

How to sign **k** bits **just once** ?

Theorem

If f is (τ, ε)-one way then Lamport's signature scheme (for k-bit messages) is $(1, \tau', 2k \cdot \varepsilon)$-EUF-CMA secure, with $\tau' = \tau + (2k-1)\mathcal{T}_{\mathsf{Eval}}$.

- In other words: If there is an Adversary $\mathcal{A}$ that chooses
  - a message $m \in \{0,1\}^k$ for Alice to legitimately authenticate
  - forges a message $m' \neq m$ with probability at least ε

  Then there is an Adversary $\mathcal{B}$ that can break the one-wayness of the function f with probability at least $\varepsilon/2k$ operates in time roughly the same as $\mathcal{A}$

How to sign **k** bits **just once** ?

Proof. $\mathcal{B}$ gets as input the description of f and $y^\star \in Y$.

- $\mathcal{B}$ picks as input an index $(i^\star, j^\star) \in \{0,1\} \times [\![1,k]\!]$
- $\mathcal{B}$ selects $2k-1$ random elements $x_{0,1}, \dots, \widehat{x_{i^\star,j^\star}}, \dots, x_{1,k} \in X$.
- $\mathcal{B}$ computes their images $y_{i,j} = f(x_{i,j}) = \mathrm{Eval}(x_{i,j})$ for $(i,j) \in \{0,1\} \times [\![1,k]\!] \setminus \{(i^\star, j^\star)\}$.
- $\mathcal{B}$ sets $y_{i^\star,j^\star} = y$
- $\mathcal{B}$ executes $\mathcal{A}$ on the public key $(y_{0,1}, y_{1,1}, \dots, y_{0,k}, y_{1,k})$
- At some point $\mathcal{A}$ query **one** message $m = m_1 \dots m_k$ to the signature oracle
  - If $m_{j^\star} = i^\star$ then $\mathcal{B}$ aborts the simulation (probability $1/2$),
  - otherwise $\mathcal{B}$ outputs a valid signature on m thanks to its knowledge of $x_{0,1}, \dots, \widehat{x_{i^\star,j^\star}}, \dots, x_{1,k}$.
- Eventually, $\mathcal{A}$ outputs a signature s' on a message $m' \neq m$ and $\mathcal{B}$ outputs $s'_{j^\star}$. The message m' differs from m in at least one position. If it is the $j^\star$-th position (probability $1/k$) and if the signature is valid (probability ε) we have $f(s'_{j^\star}) = y_{i^\star,j^\star} = y$.

$\square$

How to sign **k** bits **just once** ?

- Lamport's scheme is EUF-CMA secure assuming **only** the one-wayness of f.
- The signature generation is very efficient.

- For (generic) groups of of prime order q of n-bits, solving the discrete logarithm problem requires $2^{n/2}$ operations.
- For a 128-bit security level, we need to have a group order q of (at least) 256 bits and for an ideal $\mathbb{G}$ (an elliptic curve?), elements in $\mathbb{G}$ can be represented with 256 bits.
  The verification key is made of $256^2 = 65536$ bits and its generation requires 256 exponentiation in $\mathbb{G}$.
- The signature is made of k elements from $\mathbb{Z}_q$. The signature length is at least $256 \cdot k$ bits.
- Can sign only one message

Lamport's signatures: variants

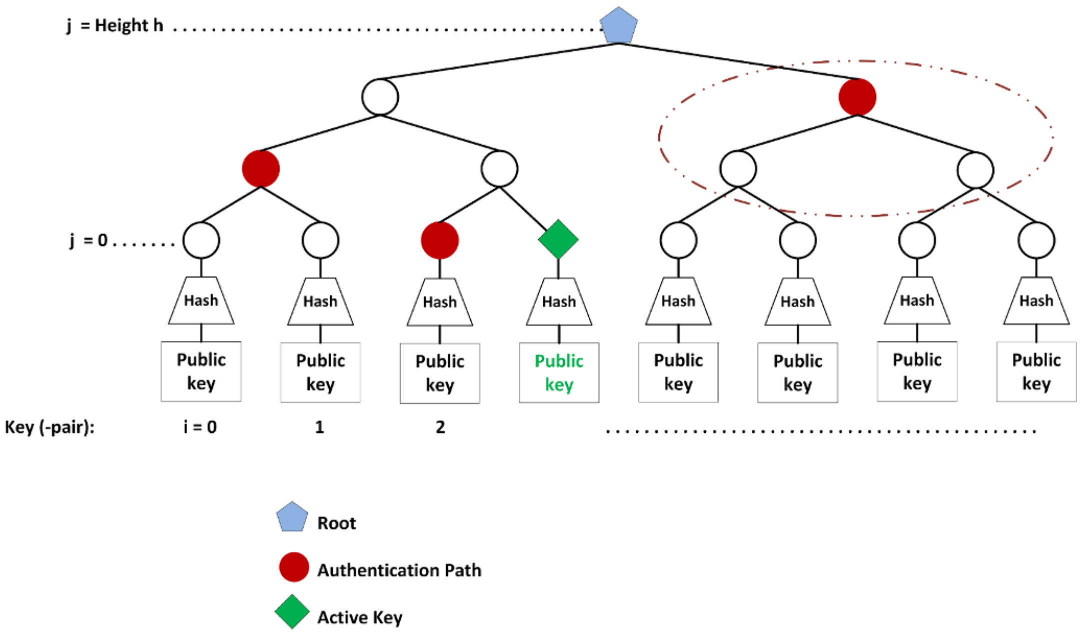
- **Short private key.** Instead of creating and storing all the random numbers of the private key a single key of sufficient size can be stored.

  The single key can then be used as the seed for a **cryptographically secure pseudorandom number generator** to create all the random numbers in the private key when needed.

- **Short public key** A Lamport signature can be combined with a hash list, making it possible to only publish a single hash instead of all the hashes in the public key.

- **Hashing the message.**
  - Unlike some other signature schemes the Lamport signature scheme **does not** require that the message m is hashed before it is signed.
  - A system for signing long messages can use a collision resistant hash function h and sign $h(m)$ instead of m.

Lamport's signatures: several messages



Groth's one-time signatures

Groth (2006)
Simulation-sound NIZK proofs for a practical language and constant size group signatures.
Advances in Cryptology - Asiacrypt 2006: pp. 444–459

Key generation: generate $vk = (X = g^x, Y = g^y, Z = g^z)$ where $x, y \xleftarrow{\$} \mathbb{Z}_p^*$

Sign: to sign $m \in \mathbb{Z}_p^*$, select $r \xleftarrow{\$} \mathbb{Z}_p^*$, compute $s = (1 - mx - yr)/z \in \mathbb{Z}_p^*$, and output $\sigma = (r, s)$.

Verify: given $\sigma \in (\mathbb{Z}_p^*)^2$, check

$$X^m Y^r Z^s = g.$$

Groth's one-time signatures

Theorem

If the discrete logarithm assumption holds in $\mathbb{G}$ then Groth's signature scheme is one-time EUF-CMA secure.

Proof idea: given a DL instance $(g, h) \in \mathbb{G}$, one sets $X = g^{a_1} h^{b_1}$, $Y = g^{a_2} h^{b_2}$, $Z = g^{a_3}$ where $a, b, c \xleftarrow{\$} \mathbb{Z}_p^*$. On signature query on m, one compute $r = -mb_1/b_2 \bmod p$ and $s = (1 - ma_1 - r_2)/a_3 \bmod p$.

Thanks to the adversary's forgery, one can retrieve the discrete logarithm of h in base g by solving a linear system. □

Graph isomorphism

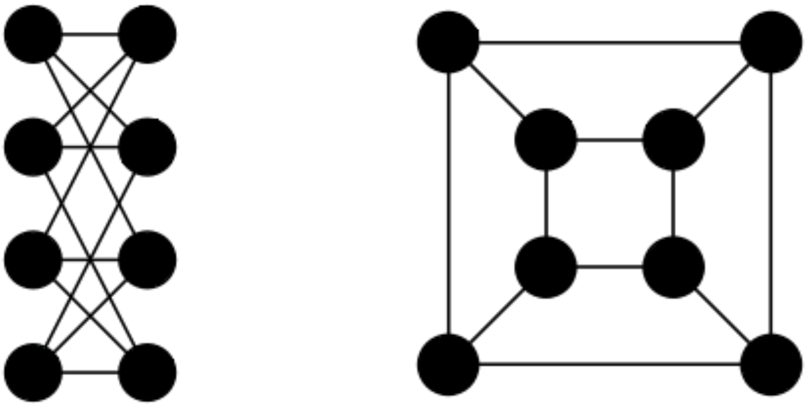
- In **graph theory**, an isomorphism of graphs G and H is a bijection between the vertex sets of G and H

$$f : V(G) \longrightarrow V(H)$$

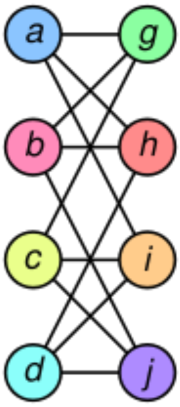
  such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H.

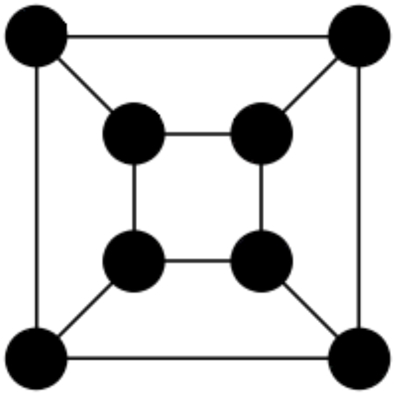
- If an isomorphism exists between two graphs, then the graphs are called isomorphic.
- The computational problem of determining whether two finite graphs are isomorphic is referred to as the **graph isomorphism problem**.
- The graph isomorphism problem is a curiosity in computational complexity theory: not known to be in $\mathcal{P}$ nor $\mathcal{NP}$-complete.

Graph isomorphism

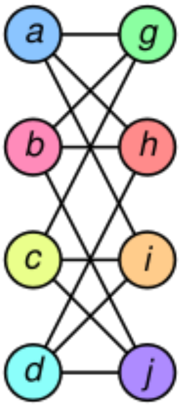


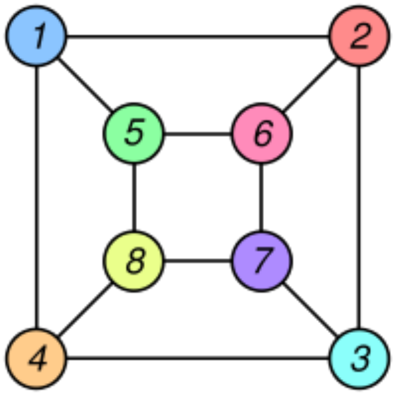
Graph isomorphism





Graph isomorphism





Aussois, Mar. 18 2019 Provable Security in PKC Damien Vergnaud 56 / 64

Zero-knowledge interactive proof

- a **zero-knowledge proof or zero-knowledge** protocol is an interactive method for one party to prove to another that a (usually mathematical) statement is true, without revealing anything other than the veracity of the statement.
- A zero-knowledge proof must satisfy three properties:
  1. **Completeness:** if the statement is true, the honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.
  2. **Soundness:** if the statement is false, no cheating prover can convince the honest verifier that it is true, except with some small probability.
  3. **Zero-knowledge:** if the statement is true, no cheating verifier learns anything other than this fact.
- The first two of these are properties of more general interactive proof systems. The third is what makes the proof zero-knowledge.

Zero-knowledge interactive proof for Graph Isomorphism

Input: Two graphs G_0 and G_1 each having vertex set $\{1, \ldots, n\}$.
Alice **knows** $\sigma \in \mathfrak{S}_n$ an isomorphism from G_0 to G_1

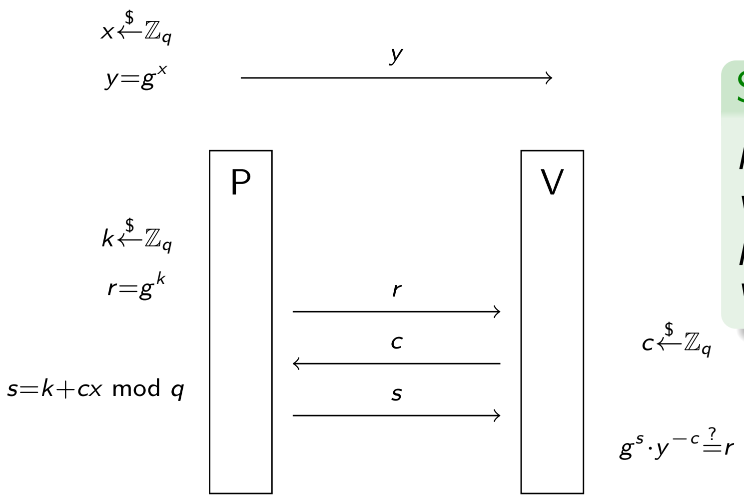
Repeat the following n times

1. Alice chooses a random permutation $\pi \in \mathfrak{S}_n$,
2. She computes H to be the image of G_0 under π and sends H to Bob,
3. Bob chooses randomly $b \in \{0, 1\}$ and sends it to Alice,
4. Alice sends $\rho = \pi \circ \sigma^b$ to Bob,
5. Bob checks if H is the image of G_b under ρ

Schnorr's ID Protocol (1989)

Let $\mathbb{G} = \langle g \rangle$ be a group of prime order q

Prover P proves to verifier V that he knows the discrete log x of a public group element $y = g^x$. It is a 3-move protocol.



Scenario

P sends $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
V sends $c \xleftarrow{\$} \mathbb{Z}_q$
P sends $s = k + cx \bmod q$
V checks whether $g^s \cdot y^{-c} = r$

The Fiat-Shamir heuristic

Fiat, Shamir (1986)
How to Prove Yourself: Practical Solutions to Identification and Signature Problems.
Advances in Cryptology - Crypto'86, Lect. Notes Comput. Science 263, pp. 186-194.

- In such a 3-pass identification scheme, the messages are called **commitment**, **challenge** and **response**. The challenge is randomly chosen by V.

Fiat-Shamir Transform: replace the challenge by a hash value taken on scheme parameters and t, thereby removing V. This transforms the protocol by making it *non-interactive*.

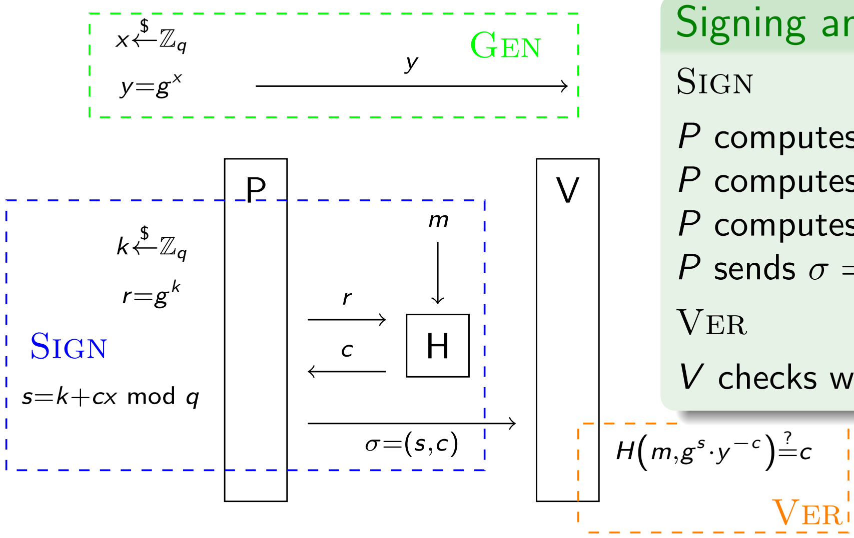
The intuition is that any "sufficiently random" hash function should preserve the security of the protocol.

(Many applications $\rightsquigarrow$ **see Damien's lectures / Luca's lectures**)

Schnorr Signatures (via the Fiat-Shamir Transform)

Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\S_i gma_H$ is a tuple of probabilistic algorithms $\S_i gma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.



Signing and Verifying

Sign

P computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
P computes $c = H(m, r)$
P computes $s = k + cx \bmod q$
P sends $\sigma = (s, c)$

Ver

V checks whether $H(m, g^s \cdot y^{-c}) = c$

Security of Schnorr Signatures - Key Only Attacks

Theorem

If there exist a $(0, \tau, \varepsilon)$-EUF-CMA adversary in the ROM (with q_H queries to the RO) against Schnorr's signature scheme (in $\mathbb{G}$), then the discrete logarithm in $\mathbb{G}$ can be solved in expected time $O(\tau \cdot q_H / \varepsilon)$.

Proof Intuition

- run the adversary $\mathcal{A}$ several times in related executions
- the process "forks" at a certain point (modification of the RO)
- hope for two executions of $\mathcal{A}$ with forgery on the same message queried to the RO (but with different hash values)
  $\rightsquigarrow$ extract the discrete logarithm

Security of Schnorr Signatures - Chosen Message Attacks

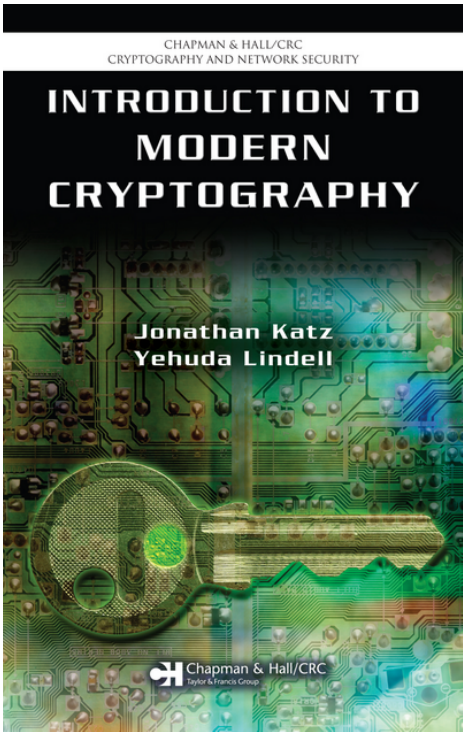
Theorem

If there exist a (q_S, τ, ε)-EUF-CMA adversary in the ROM (with q_H queries to the RO) against Schnorr's signature scheme (in $\mathbb{G}$), then the discrete logarithm in $\mathbb{G}$ can be solved in expected time $O(\tau \cdot q_H/\varepsilon)$.

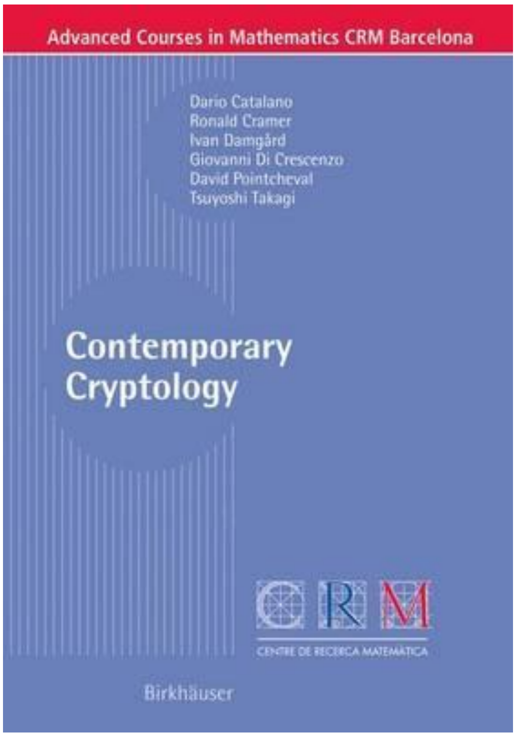
The previous result can be adapted readily for an EF-CMA adversary.

In order to answer signing queries, one simply uses the simulator of the zero-knowledge proof (r, h, s), and we set $H(m, r) := h$. The random oracle programming may fail, but with negligible probability.

References



References


Advanced Courses in Mathematics CRM Barcelona
Dario Catalano
Ronald Cramer
Ivan Damgård
Giovanni Di Crescenzo
David Pointcheval
Tsuyoshi Takagi
Contemporary Cryptology
CENTRE DE RECERCA MATEMÀTICA
Birkhäuser